DBMS_LOGMNR

The `DBMS_LOGMNR` package, one of a set of LogMiner packages, contains the subprograms you use to initialize the LogMiner tool and to begin and end a LogMiner session.

> **See Also:**
>
> *Oracle Database Utilities* for information regarding LogMiner.

This chapter contains the following topics:

Using DBMS_LOGMNR

This section contains the following topics, which relate to using the `DBMS_LOGMNR` package:

Overview

Oracle LogMiner, which is part of Oracle Database, enables you to query online and archived redo log files through a SQL interface. The `DBMS_LOGMNR` package provides the majority of the tools needed to start and stop LogMiner and specify the redo log files of interest.

All changes made to user data or to the database dictionary are recorded in the Oracle redo log files so that database recovery operations can be performed. You can take advantage of the data recorded in the redo log files to accomplish other tasks, such as:

- Pinpointing when a logical corruption to a database, such as errors made at the application level, may have begun
- Determining what actions you would have to take to perform fine-grained recovery at the transaction level.
- Performance tuning and capacity planning through trend analysis.
- Track any data manipulation language (DML) and data definition language (DDL) statements executed on the database, the order in which they were executed, and who executed them.

See Also:

Chapter 55, "DBMS_LOGMNR_D"for information on the package subprograms that extract a LogMiner dictionary and re-create LogMiner tables in alternate tablespaces

Security Model

You must have the `EXECUTE_CATALOG_ROLE` role to use the `DBMS_LOGMNR` package.

Constants

The DBMS_LOGMNR package defines several enumerated constants for specifying parameter values. Enumerated constants must be prefixed with the package name, for example, DBMS_LOGMNR.NEW.

Table 54-1describes the constants for the ADD_LOGFILE options flag in the DBMS_LOGMNR package.

Table 54-1 Constants for ADD_LOGFILE Options Flag

Constant	Description
NEW	Implicitly calls the DBMS_LOGMNR.END_LOGMNR procedure to end the current LogMiner session and then creates a new session. The new session starts a new list of redo log files to be analyzed, beginning with the redo log file you specify.
ADDFILE	Adds the specified redo log file to the list of redo log files to be analyzed. Any attempt to add a duplicate file raises an exception (ORA-01289). This is the default if no options flag is specified.

Table 54-2describes the constants for the START_LOGMNR options flag in the DBMS_LOGMNR package.

Table 54-2 Constants for START_LOGMNR Options Flag

Constant	Description

Constant	Description
COMMITTED_DATA_ONLY	If set, DML statements corresponding to committed transactions are returned. DML statements corresponding to a committed transaction are grouped together. Transactions are returned in their commit order. Transactions that are rolled back or in-progress are filtered out, as are internal redo records (those related to index operations, management, and so on). If this option is not set, all rows for all transactions (committed, rolled back, and in-progress) are returned in the order in which they are found in the redo logs (in order of SCN values).
SKIP_CORRUPTION	Directs a select operation on the V$LOGMNR_CONTENTS view to skip any corruptions in the redo log file being analyzed and continue processing. This option works only when a block in the redo log file (and not the header of the redo log file) is corrupt. You should check the INFO column in the V$LOGMNR_CONTENTS view to determine the corrupt blocks skipped by LogMiner. When a corruption in the redo log file is skipped, the OPERATION column contains the value CORRUPTED_BLOCKS, and the STATUS column contains the value 1343.
DDL_DICT_TRACKING	If the LogMiner dictionary in use is a flat file or in the redo log files, LogMiner updates its internal dictionary if a DDL event occurs. This ensures that correct SQL_REDO and SQL_UNDO information is maintained for objects that are modified after the LogMiner internal dictionary is built. The database to which LogMiner is connected must be open. This option cannot be used in conjunction with the DICT_FROM_ONLINE_CATALOG option and cannot be used when the LogMiner dictionary being used is one that was extracted to a flat file prior to Oracle9*i*.

Constant	Description
DICT_FROM_ONLINE_CATALOG	Directs LogMiner to use the current online database dictionary rather than a LogMiner dictionary contained in a flat file or in the redo log files being analyzed. This option cannot be used in conjunction with the DDL_DICT_TRACKING option. The database to which LogMiner is connected must be the same one that generated the redo log files. Expect to see a value of 2 in the STATUS column of the V$LOGMNR_CONTENTSview if the table definition in the database does not match the table definition in the redo log file.
DICT_FROM_REDO_LOGS	If set, LogMiner expects to find a LogMiner dictionary in the redo log files that were specified. The redo log files are specified with the DBMS_LOGMNR.ADD_LOGFILE procedure or with the DBMS_LOGMNR.START_LOGMNR procedure with the CONTINUOUS_MINE option.
NO_SQL_DELIMITER	If set, the SQL delimiter (a semicolon) is not placed at the end of reconstructed SQL statements. This is helpful for applications that open a cursor and then execute the reconstructed statements.
NO_ROWID_IN_STMT	If set, the ROWID clause is not included in the reconstructed SQL statements. The redo log file may already contain logically unique identifiers for modified rows if supplemental logging is enabled. When using this option, you must be sure that supplemental logging was enabled in the source database at the appropriate level and that no duplicate rows exist in the tables of interest. LogMiner does not make any guarantee regarding the uniqueness of logical row identifiers.

Constant	Description
PRINT_PRETTY_SQL	If set, LogMiner formats the reconstructed SQL statements for ease of reading. These reconstructed SQL statements are not executable.
CONTINUOUS_MINE	Directs LogMiner to automatically add redo log files, as needed, to find the data of interest. You only need to specify the first log to start mining, or just the starting SCN or date to indicate to LogMiner where to begin mining logs. You are not required to specify any redo log files explicitly. LogMiner automatically adds and mines the (archived and online) redo log files for the data of interest. This option requires that LogMiner is connected to the same database instance that is generating the redo log files. It also requires that the database be mounted and that archiving be enabled. Beginning with Oracle Database release 10.1, the CONTINUOUS_MINE options is supported for use in an Oracle Real Application Clusters environment.

Views

The DBMS_LOGMNR package uses the views listed in the section on Accessing LogMiner Operational Information in Views in *Oracle Database Utilities*.

Operational Notes

A **LogMiner session** begins with a call to DBMS_LOGMNR.ADD_LOGFILE or DBMS_LOGMNR.START_LOGMNR (the former if you plan to specify log files explicitly; the latter if you plan to use continuous mining). The session ends with a call to DBMS_LOGMNR.END_LOGMNR. Within a LogMiner session, you can specify the redo log files to be analyzed and the SCN or time range of interest; then you can issue SQL SELECT statements against the V$LOGMNR_CONTENTS view to retrieve the data of interest.

Summary of DBMS_LOGMNR Subprograms

Table 54-3 DBMS_LOGMNR Package Subprograms

Subprogram	Description
ADD_LOGFILE Procedure	Adds a redo log file to the existing or newly created list of redo log files for LogMiner to process, so that if a new list is created, this marks the beginning of a LogMiner session
COLUMN_PRESENT Function	Call this function for any row returned from the `V$LOGMNR_CONTENTS` view to determine if undo or redo column values exist for the column specified by the `column_name` input parameter to this function
END_LOGMNR Procedure	Finishes a LogMiner session
MINE_VALUE Function	Call this function for any row returned from the `V$LOGMNR_CONTENTS` view to retrieve the undo or redo column value of the column specified by the `column_name` input parameter to this function
REMOVE_LOGFILE Procedure	Removes a redo log file from the list of redo log files for LogMiner to process
START_LOGMNR Procedure	Initializes the LogMiner utility and starts LogMiner (unless the session was already started with a call to `DBMS_LOGMNR.ADD_LOGFILE`)

ADD_LOGFILE Procedure

This procedure adds a file to an existing or newly created list of log files for LogMiner to process.

Syntax

```
DBMS_LOGMNR.ADD_LOGFILE ( LogFileName IN VARCHAR2, options IN BINARY_INTEGER default ADDFILE );
```

Parameters

Table 54-4 ADD_LOGFILE Procedure Parameters

Parameter	Description
`LogFileName`	Specifies the name of the redo log file to add to the list of redo log files to be analyzed during this session.
`options`	Does one of the following: • Starts a new LogMiner session and a new list of redo log files for analysis (`DBMS_LOGMNR.NEW`) • Adds a file to an existing list of redo log files for analysis (`DBMS_LOGMNR.ADDFILE`) See Table 54-1, "Constants for ADD_LOGFILE Options Flag".

Exceptions

Table 54-5 ADD_LOGFILE Procedure Exceptions

Exception	Description

Exception	Description
ORA-01284	Specified file cannot be opened.
ORA-01287	Specified file is from a different database incarnation.
ORA-01289	Specified file has already been added to the list. Duplicate redo log files cannot be added.
ORA-01290	Specified file is not in the current list and therefore cannot be removed from the list.
ORA-01324	Specified file cannot be added to the list because there is a DB_ID mismatch.

Usage Notes

- Before querying the V$LOGMNR_CONTENTS view, you must make a successful call to the DBMS_LOGMNR.START_LOGMNR procedure (within the current LogMiner session).
- Unless you specify the CONTINUOUS_MINE option, the LogMiner session must be set up with a list of redo log files to be analyzed. Use the ADD_LOGFILE procedure to specify the list of redo log files to analyze.
- If you are not using the CONTINUOUS_MINE option and you want to analyze more than one redo log file, you must call the ADD_LOGFILE procedure separately for each redo log file. The redo log files do not need to be registered in any particular order.
- Both archived and online redo log files can be mined.

- After you have added the first redo log file to the list, each additional redo log file that you add to the list must be associated with the same database and database `RESETLOGS SCN` as the first redo log file. (The database `RESETLOGS SCN` uniquely identifies each execution of an `ALTER DATABASE OPEN RESETLOGS` statement. When the online redo logs are reset, Oracle creates a new and unique incarnation of the database.)

- To analyze the redo log files from a different database (or a database incarnation with a different database `RESETLOGS SCN`) than that with which the current list of redo log files is associated, use the `END_LOGMNR` procedure to end the current LogMiner session, and then build a new list using the `ADD_LOGFILE` procedure.

- LogMiner matches redo log files by the log sequence number. Thus, two redo log files with different names but with the same log sequence number will return the ORA-01289 exception. For instance, the online counterpart of an archived redo log file has a different name from the archived redo log file, but attempting to register it with LogMiner after registering the archived counterpart will result in the ORA-01289 exception being returned.

COLUMN_PRESENT Function

This function is designed to be used in conjunction with the `MINE_VALUE` function.

If the `MINE_VALUE` function returns a `NULL` value, it can mean either:

- The specified column is not present in the redo or undo portion of the data.

- The specified column is present and has a `NULL` value.

To distinguish between these two cases, use the `COLUMN_PRESENT` function, which returns a `1` if the column is present in the redo or undo portion of the data. Otherwise, it returns a `0`.

Syntax

```
DBMS_LOGMNR.COLUMN_PRESENT ( sql_redo_undo IN RAW, column_name IN VARCHAR2 default '') RETURN NUMBER;
```

Parameters

Table 54-6 COLUMN_PRESENT Function Parameters

Parameter	Description
sql_redo_undo	Specifies either the REDO_VALUE or the UNDO_VALUE column in the V$LOGMNR_CONTENTS view from which to extract data values. See the Usage Notes for more information.
column_name	Specifies the fully qualified name (schema.table.column) of the column for which this function will return information.

Return Values

Table 54-7describes the return values for the COLUMN_PRESENT function. The COLUMN_PRESENT function returns 1 if the self-describing record (the first parameter) contains the column specified in the second parameter. This can be used to determine the meaning of NULL values returned by the DBMS_LOGMNR.MINE_VALUE function.

Table 54-7 Return Values for COLUMN_PRESENT Function

Return	Description
0	Specified column is not present in this row of V$LOGMNR_CONTENTS.
1	Column is present in this row of V$LOGMNR_CONTENTS.

Exceptions

Table 54-8 COLUMN_PRESENT Function Exceptions

Exception	Description

Exception	Description
ORA-01323	Currently, a LogMiner dictionary is not associated with the LogMiner session. You must specify a LogMiner dictionary for the LogMiner session.
ORA-00904	Value specified for the column_name parameter is not a fully qualified column name.

Usage Notes

- To use the COLUMN_PRESENT function, you must have successfully started LogMiner.
- The COLUMN_PRESENT function must be invoked in the context of a select operation on the V$LOGMNR_CONTENTS view.
- The COLUMN_PRESENT function does not support LONG, LOB, ADT, or COLLECTION datatypes.
- The value for the sql_redo_undo parameter depends on the operation performed and the data of interest:
  - If an update operation was performed and you want to know what the value was prior to the update operation, specify UNDO_VALUE.
  - If an update operation was performed and you want to know what the value is after the update operation, specify REDO_VALUE.
  - If an insert operation was performed, typically you would specify REDO_VALUE (because the value of a column prior to an insert operation will always be NULL).
  - If a delete operation was performed, typically you would specify UNDO_VALUE (because the value of a column after a delete operation will always be NULL).

END_LOGMNR Procedure

This procedure finishes a LogMiner session. Because this procedure performs cleanup operations that may not otherwise be done, you must use it to properly end a LogMiner session. This procedure is called automatically when you log out of a database session or when you call `DBMS_LOGMNR.ADD_LOGFILE` and specify the `NEW`option.

Syntax

```
DBMS_LOGMNR.END_LOGMNR;
```

Exceptions

Table 54-9 END_LOGMNR Procedure Exception

Exception	Description
`ORA-01307`	No LogMiner session is currently active. The `END_LOGMNR` procedure was called without adding any log files or before the `START_LOGMNR` procedure was called

MINE_VALUE Function

This function facilitates queries based on a column's data value. This function takes two arguments. The first one specifies whether to mine the redo (`REDO_VALUE`) or undo (`UNDO_VALUE`) portion of the data. The second argument is a string that specifies the fully qualified name of the column to be mined. The `MINE_VALUE` function always returns a string that can be converted back to the original datatype.

Syntax

```
DBMS_LOGMNR.MINE_VALUE ( sql_redo_undo IN RAW, column_name IN VARCHAR2 default '') RETURN VARCHAR2;
```

Parameters

Table 54-10 MINE_VALUE Function Parameters

Parameter	Description
sql_redo_undo	Specifies either the REDO_VALUE or the UNDO_VALUE column in the V$LOGMNR_CONTENTS view from which to extract data values. See the Usage Notes for more information.
column_name	Specifies the fully qualified name (schema.table.column) of the column for which this function will return information.

Return Values

Table 54-11 Return Values for MINE_VALUE Function

Return	Description
NULL	The column is not contained within the self-describing record, or the column value is NULL. To distinguish between the two different null possibilities, use the DBMS_LOGMNR.COLUMN_PRESENT function.
NON-NULL	The column is contained within the self-describing record; the value is returned in string format.

Exceptions

Table 54-12 MINE_VALUE Function Exceptions

Exception	Description

Exception	Description
ORA-01323	Invalid state. Currently, a LogMiner dictionary is not associated with the LogMiner session. You must specify a LogMiner dictionary for the LogMiner session.
ORA-00904	Invalid identifier. The value specified for the column_name parameter was not a fully qualified column name.

Usage Notes

- To use the MINE_VALUE function, you must have successfully started LogMiner.
- The MINE_VALUE function must be invoked in the context of a select operation from the V$LOGMNR_CONTENTS view.
- The MINE_VALUE function does not support LONG, LOB, ADT, or COLLECTION datatypes.
- The value for the sql_redo_undo parameter depends on the operation performed and the data of interest:
  - If an update operation was performed and you want to know what the value was prior to the update operation, specify UNDO_VALUE.
  - If an update operation was performed and you want to know what the value is after the update operation, specify REDO_VALUE.
  - If an insert operation was performed, typically you would specify REDO_VALUE (because the value of a column prior to an insert operation will always be null).
  - If a delete operation was performed, typically you would specify UNDO_VALUE (because the value of a column after a delete operation will always be null).

REMOVE_LOGFILE Procedure

This procedure removes a redo log file from an existing list of redo log files for LogMiner to process.

> **Note:**
> This procedure replaces the `REMOVEFILE` constant that was an option on the `ADD_LOGFILE` procedure prior to Oracle Database 10*g*.

Syntax

```
DBMS_LOGMNR.REMOVE_LOGFILE ( LogFileName IN VARCHAR2);
```

Parameters

Table 54-13 REMOVE_LOGFILE Procedure Parameters

Parameter	Description
`LogFileName`	Specifies the name of the redo log file to be removed from the list of redo log files to be analyzed during this session.

Exceptions

Table 54-14 REMOVE_LOGFILE Procedure Exception

Exception	Description
`ORA-01290`	Cannot remove unlisted log file

Usage Notes

- Before querying the `V$LOGMNR_CONTENTS` view, you must make a successful call to the `DBMS_LOGMNR.START_LOGMNR` procedure (within the current LogMiner session).

- You can use this procedure to remove a redo log file from the list of redo log files for LogMiner to process if you know that redo log file does not contain any data of interest.
- Multiple redo log files can be removed by calling this procedure repeatedly.
- The redo log files do not need to be removed in any particular order.
- To start a new list of redo log files for analysis, use the `END_LOGMNR` procedure to end the current LogMiner session, and then build a new list using the `ADD_LOGFILE`procedure.
- Even if you remove all redo log files from the list, any subsequent calls you make to the `ADD_LOGFILE` procedure must match the database ID and `RESETLOGS SCN` of the removed redo log files. Therefore, to analyze the redo log files from a different database (or a database incarnation with a different database `RESETLOGS SCN`) than that with which the current list of redo log files is associated, use the `END_LOGMNR` procedure to end the current LogMiner session, and then build a new list using the `ADD_LOGFILE` procedure.

START_LOGMNR Procedure

This procedure starts LogMiner by loading the dictionary that LogMiner will use to translate internal schema object identifiers to names.

Syntax

```
DBMS_LOGMNR.START_LOGMNR ( startScn IN NUMBER default 0, endScn IN NUMBER default 0, startTime IN DATE default '01-jan-1988',
endTime IN DATE default '31-dec-2110', DictFileName IN VARCHAR2 default '', Options IN BINARY_INTEGER default 0 );
```

Parameters

Table 54-15 START_LOGMNR Procedure Parameters

Parameter	Description

Parameter	Description
startScn	Directs LogMiner to return only redo records with an SCN greater than or equal to the startScn specified. This fails if there is no redo log file containing the specified startScn value. (You can query the FILENAME, LOW_SCN, and NEXT_SCN columns in the V$LOGMNR_LOGS view for each redo log file to determine the range of SCN values contained in each redo log file.)
endScn	Directs LogMiner to return only redo records with an SCN less than or equal to the endScn specified. If you specify an endScn value that is beyond the value in any redo log file, then LogMiner will use the greatest endScn value in the redo log file that contains the most recent changes. (You can query the FILENAME,LOW_SCN, and NEXT_SCN columns in the V$LOGMNR_LOGS view for each redo log file to determine the range of SCN values contained in each redo log file.)
startTime	Directs LogMiner to return only redo records with a timestamp greater than or equal to the startTime specified. This fails if there is no redo log file containing the specified startTime value. (You can query the FILENAME, LOW_TIME, and HIGH_TIME columns in the V$LOGMNR_LOGS view for each redo log file to determine the range of time covered in each redo log file.) This parameter is ignored if startScn is specified. See the Usage Notes for additional information.

Parameter	Description
endTime	Directs LogMiner to return only redo records with a timestamp less than or equal to the endTime specified. If you specify an endTime value that is beyond the value in any redo log file, then LogMiner will use the greatest endTime in the redo log file that contains the most recent changes. You can query the FILENAME, LOW_TIME, and HIGH_TIME columns in the V$LOGMNR_LOGS view for each redo log file to determine the range of time covered in each redo log file.) This parameter is ignored if endScn is specified. See the Usage Notes for additional information.
DictFileName	Specifies the flat file that contains the LogMiner dictionary. It is used to reconstruct SQL_REDO and SQL_UNDO columns in V$LOGMNR_CONTENTS, as well as to fully translate SEG_NAME, SEG_OWNER, SEG_TYPE_NAME, TABLE_NAME, and TABLE_SPACE columns. The fully qualified path name for the LogMiner dictionary file must be specified. (This file must have been created previously through the DBMS_LOGMNR_D.BUILD procedure.) You need to specify this parameter only if neither DICT_FROM_REDO_LOGS nor DICT_FROM_ONLINE_CATALOG is specified.
options	See Table 54-2, "Constants for START_LOGMNR Options Flag".

Exceptions

Table 54-16 START_LOGMNR Procedure Exceptions

Exception	Description

Exception	Description
ORA-01280	Internal error encountered.
ORA-01281	`startScn` or `endScn` parameter value is not a valid SCN, or `endScn` is less than `startScn`.
ORA-01282	value for the `startTime` parameter was greater than the value specified for the `endTime` parameter, or there was no redo log file that was compatible with the date range specified with the `startTime` and `endTime` parameters.
ORA-01283	Options parameter specified is invalid.
ORA-01284	LogMiner dictionary file specified in the `DictFileName` parameter has a full path length greater than 256 characters, or the file cannot be opened.
ORA-01285	Error reading specified file.
ORA-01291	Redo log files that are needed to satisfy the user's requested SCN or time range are missing.
ORA-01292	No log file has been specified for the current LogMiner session.
ORA-01293	Mounted database required for specified LogMiner options.
ORA-01294	Error occurred while processing information in the specified dictionary file, possible corruption.

Exception	Description
ORA-01295	Specified LogMiner dictionary does not correspond to the database that produced the log files being analyzed.
ORA-01296	Character set mismatch between specified LogMiner dictionary and log files.
ORA-01297	Redo version mismatch between LogMiner dictionary and log files.
ORA-01299	Specified LogMiner dictionary corresponds to a different database incarnation.
ORA-01300	Writable database required for specified LogMiner options.

Usage Notes

- LogMiner can use a dictionary that you previously extracted to the redo log files or to a flat file, or you can specify that LogMiner use the online catalog if LogMiner is mining data from the source system. See *Oracle Database Utilities* and Chapter 55, "DBMS_LOGMNR_D" in this manual for more information about the LogMiner dictionary.
- After executing the START_LOGMNR procedure, you can query the following views:
  - V$LOGMNR_CONTENTS - contains history of information in redo log files
  - V$LOGMNR_DICTIONARY - contains current information about the LogMiner dictionary file extracted to a flat file
  - V$LOGMNR_PARAMETERS - contains information about the LogMiner session

  (You can query the V$LOGMNR_LOGS view after a redo log file list has been added to the list of files that LogMiner is to mine.)

- Parameters and options are not persistent across calls to `DBMS_LOGMNR.START_LOGMNR`. You must specify all desired parameters and options (including SCN and time ranges) each time you call `DBMS_LOGMNR.START_LOGMNR`

- Be aware that specifying redo log files using a timestamp is not precise.

- The `CONTINUOUS_MINE` option directs LogMiner to automatically add redo log files, as needed, to find the data of interest. You need to specify only the first log to start mining, or just the starting SCN or date to indicate to LogMiner where to begin mining logs. Keep the following in mind when using the `CONTINUOUS_MINE`option:

  - The database control file will hold information about a limited number of archived redo log files, although the number of entries can be quite large. Query the `V$ARCHIVED_LOGS` view to determine which redo log file entries will be found by LogMiner.

    Even if an entry is listed in the database control file (and the `V$ARCHIVED_LOGS` view), the archived redo log file may not be accessible by LogMiner for various reasons. For example, the archived redo log file may have been deleted or moved from its location (maybe because of a backup operation to tape), or the directory where it resides may not be not available.

  - If you specify the `CONTINUOUS_MINE` option and an ending time or SCN that will occur in the future (or you do not specify an end time or SCN), a query of the `V$LOGMNR_CONTENTS` view will not finish until the database has generated redo log files beyond the specified time or SCN. In this scenario, LogMiner will automatically add archived redo log files to the LogMiner redo log file list as they are generated. In addition, in this scenario only, LogMiner may automatically remove redo log files from the list to keep it at 50 processed redo files. This is to save PGA memory as LogMiner automatically adds redo log files to the list. If LogMiner did not perform automated removal, memory could eventually be exhausted.

  - LogMiner can mine online redo logs. However, if the `CONTINUOUS_MINE` option is not specified, it is possible that the database is writing to the online redo log file at the same time that LogMiner is reading the online redo log file. If a log switch occurs while LogMiner is reading an online redo log file, the database will overwrite what LogMiner is attempting to read. The data that LogMiner returns if the file it is trying to read gets overwritten by the database is unpredictable.

- Keep the following in mind regarding starting and ending times or SCN ranges:

  - If you specify neither a `startTime` nor a `startScn` parameter, LogMiner will set the `startScn` parameter to use the lowest SCN value from the redo log file that contains the oldest changes.

  - If you specify both time and SCN values, LogMiner uses the SCN value or values and ignores the time values.

- If you specify starting and ending time or SCN values and they are found in the LogMiner redo log file list, then LogMiner mines the logs indicated by those values.
- If you specify starting and ending times or SCN values that are not in the LogMiner redo log file list, and you specify `DBMS_LOGMNR.START_LOGMNR` without the `CONTINUOUS_MINE` option, and you specify:
  - `0` for the `startTime` or `startScn` value, then the lowest SCN in the LogMiner redo log file list will be used as the `startScn`
  - A nonzero number for the `startTime` or `startScn` value, then an error is returned
  - `0` or a nonzero number for the `endTime` or `endScn` value, then the highest SCN in the LogMiner redo log file list will be used as the `endScn`
- If you specify starting and ending times or SCN values and they are not found in the LogMiner redo log file list, and you specify `DBMS_LOGMNR.START_LOGMNR` with the `CONTINUOUS_MINE` option, and you specify:
  - `0` for the `startTime` or `startScn` value, then an error is returned.
  - A `startTime` or `startScn` value that is greater than any value in the database's archived redo log files, then LogMiner starts mining in the online redo log file. LogMiner will continue to process the online redo log file until it finds a change at, or beyond, the requested starting point before it returns rows from the `V$LOGMNR_CONTENTS` view.
  - An `endTime` or `endScn` parameter value that indicates a time or SCN in the future, then LogMiner includes the online redo log files when it mines. When you query the `V$LOGMNR_CONTENTS` view, rows will be returned from this view as changes are made to the database, and will not stop until LogMiner sees a change beyond the requested ending point.
  - `0` for the `endTime` or `endScn` parameter value, then LogMiner includes the online redo log files when it mines. When you query the `V$LOGMNR_CONTENTS` view, rows will be returned from this view as changes are made to the database, and will not stop until you enter CTL+C or you terminate the PL/SQL cursor.